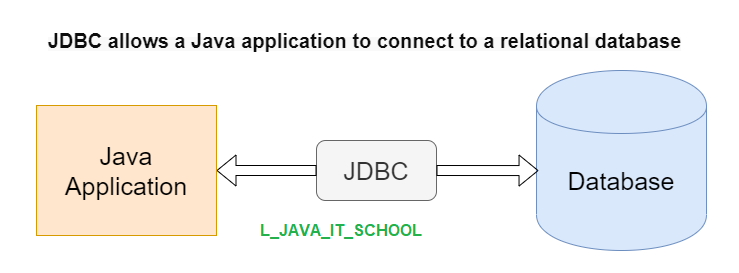


**What is JDBC?**

Java Database Connectivity or JDBC API provides industry-standard and database-independent connectivity between the Java applications and relational database servers (relational databases, spreadsheets, and flat files).

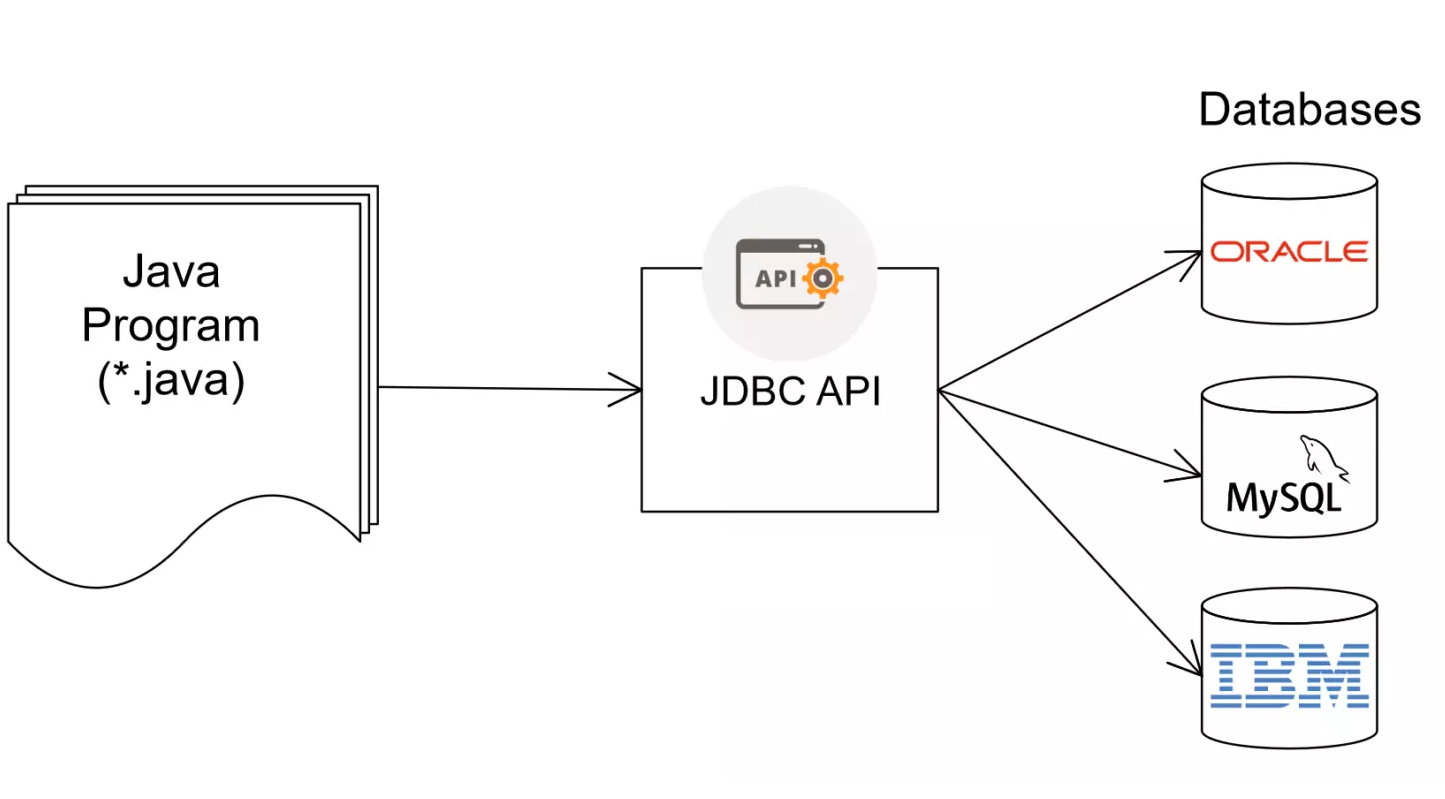
JDBC stands for Java DataBase Connectivity. It is an API for interacting with the databases using SQL( Structured Query Language ). We can retrieve, insert, delete and update data using Java code.

JDBC API is a collection of classes, interfaces, and other object-oriented artifacts grouped in JDBC packages.

****

**JDBC allows a Java application to connect to a relational database**. The major databases are supported such as Oracle, Microsoft SQL Server, DB2 and many others.

JDBC helps you to write Java applications that manage these three programming activities:

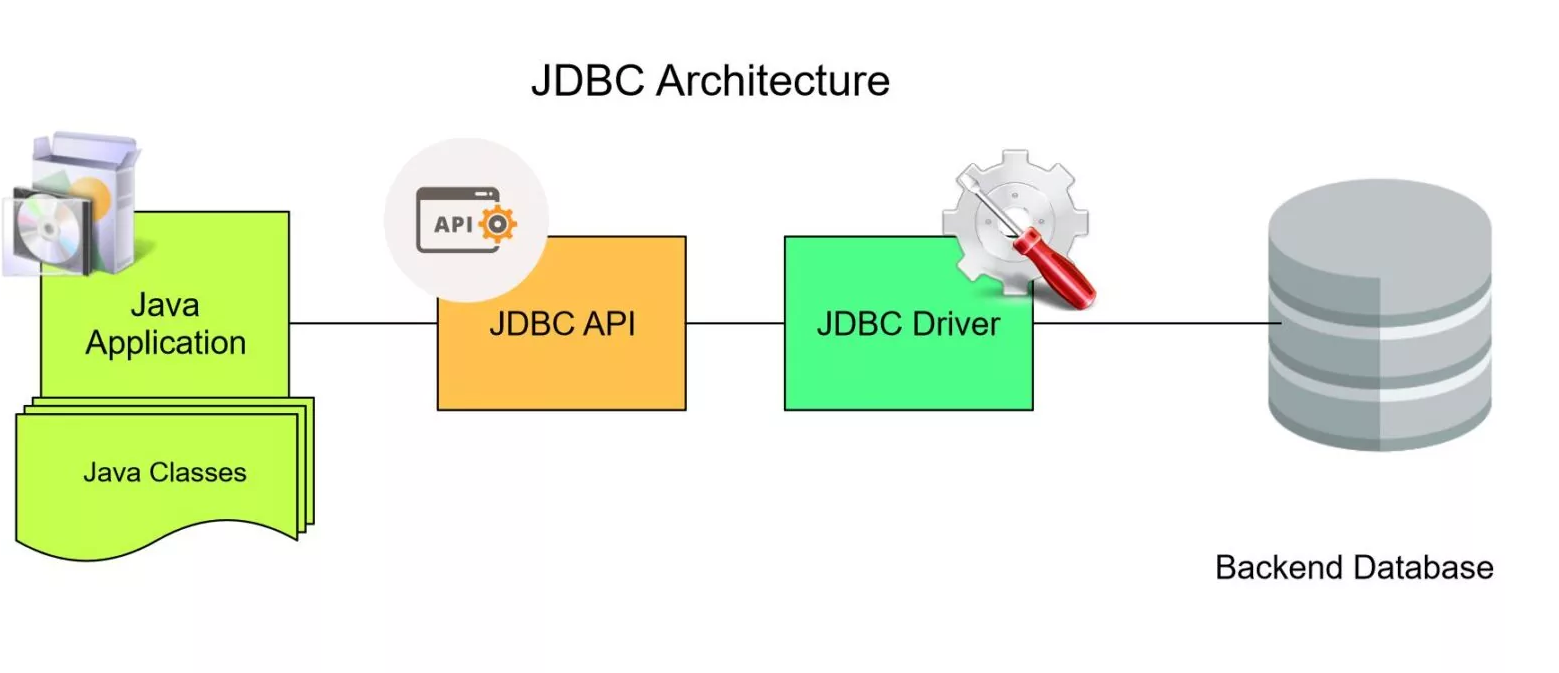


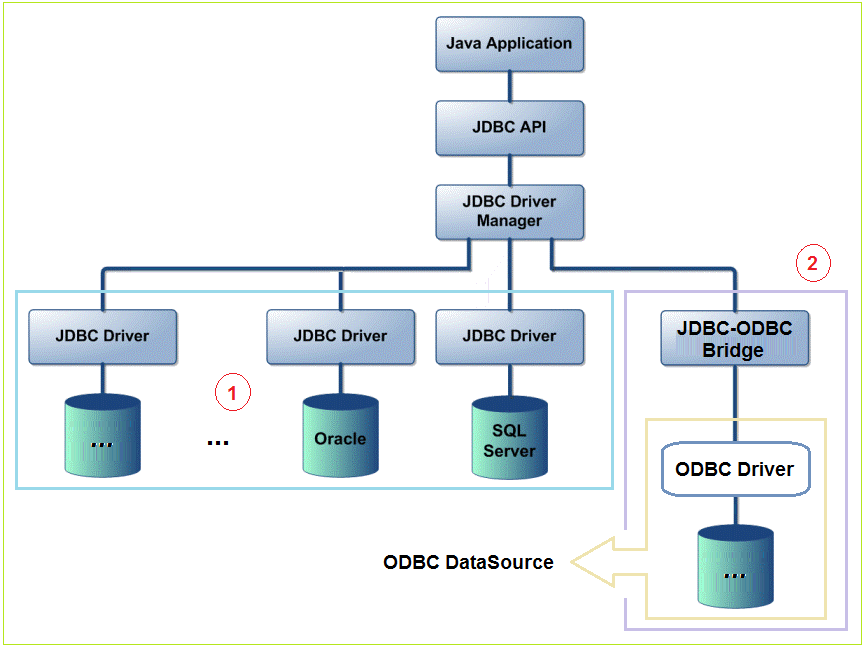
## **JDBC Architecture**

The major components are as follows:

* Java Application
* JDBC API
* JDBC Driver
* Database

A high-level JDBC Architecture is shown below:





* Oracle
* MySQL
* SQLServer

|  |  |
| --- | --- |
| Database | Library |
| Oracle | ojdbc6.jar, ojdbc8.jar, ojdbc14.jar |
| MySQL | mysql-connector-java-x.jar |
| SQL Server | jtds-x.jar |
| sqljdbc4.jar |

## **JDBC API Versions**

There are several versions of JDBC API

* JDBC 1.x
* JDBC 2.x
* JDBC 3.x
* JDBC 4.x

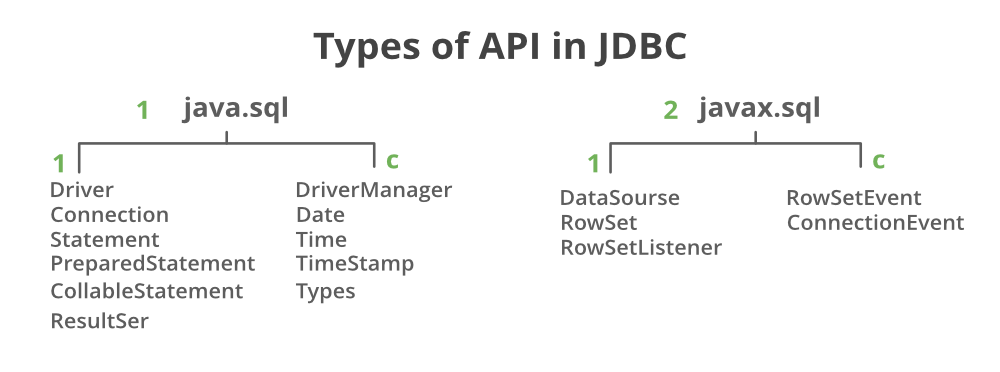
## **JDBC Packages**

JDBC includes the following JDBC packages

* java.sql
* javax.sql

These packages contain the necessary classes and interfaces for developing the JDBC application.

We automatically get both packages when you download the Java Platform Standard Edition (Java SE) 8.9 11-16 18 19 AND ALL LETTERS JDK Jdbc API consists of two packages java.sql package javax.sql package



## Classes In Jdbc

* DriverManager
* SQLException
* Types
* Date
* Time

## Interfaces In Jdbc

* Connection
* Statement
* PreparedStatement
* CallableStatementResultset
* ResultSetMetaData
* DatabaseMetaData
* Driver
* Blob
* Clob

# JDBC Driver Types

 JDBC API with a particular database management system(MySQL, Oracle, etc), we need a JDBC technology-based driver to mediate between JDBC technology and the database. Depending on various factors, a driver might be written purely in the Java programming language or in a mixture of the Java programming language and Java Native Interface (JNI) native methods.

* Type 1 –  JDBC-ODBC Bridge Driver
* Type 2 – Native API partly Java Driver
* Type 3 – Middleware Driver
* Type 4 – Pure Java Driver thin

## Type 4: All Java driver

Type 4: Drivers that are pure Java and implement the network protocol for a specific data source. The client connects directly to the data source.

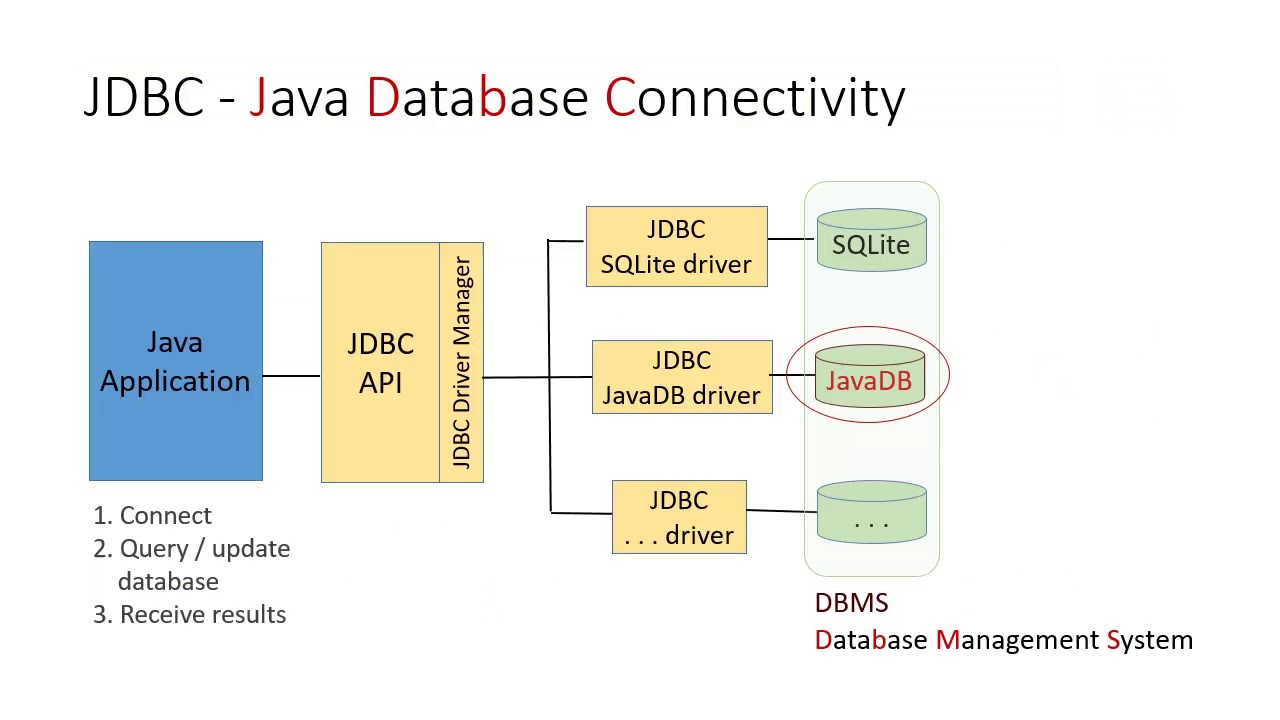
Type 4 drivers are pure 100% Java and implement the network protocol for a specific data source. The JDBC client connects directly to the data source. These drivers are platform independent and can be used in Java applications.

Example:

Oracle JDBC Thin Driver is a Type 4 driver. Oracle JDBC driver can be client-side or server-side. Oracle thin driver is stateful.

Oracale-ojdbc6.jar. ojdbc8.jar. ojdbc14.jar.

MySQL Connector/J is a Type 4 driver.

**[](https://4.bp.blogspot.com/-OIiezx01y7M/W78-Ort5u4I/AAAAAAAAEPY/a7zeUmu2McsUzoqS5_wSf_1jQmYFRCZHACLcBGAs/s1600/jdbc.jpg)**

# Java Database Connectivity with 5 Steps

There are 5 steps to connect any java application with the database using JDBC. These steps are as follows:

* Register the Driver class
* Create connection
* Create statement
* Execute queries
* Close connection



1. Driver Software
2. Statement object
3. ResultSet
4. Connection object
5. SQL query

| **Database** | **Driver name** |
| --- | --- |
| MySQL | com.mysql.jdbc.Driver |

ORACALE oracle.jdbc.OracleDriver

### **1) Register the driver class**

|  |
| --- |
| The **forName()** method of Class class is used to register the driver class. This method is used to dynamically load the driver class. |

### **Syntax of forName() method**

**public** **static** **void** forName(String className)**throws** ClassNotFoundException

Class.forName("oracle.jdbc.driver.OracleDriver");

### **2) Create the connection object**

|  |
| --- |
| The **getConnection()** method of DriverManager class is used to establish connection with the database. |

### **Syntax of getConnection() method**

1) **public** **static** Connection getConnection(String url)**throws** SQLException

2) **public** **static** Connection getConnection(String url,String name,String password)

**throws** SQLException

1. Connection con=DriverManager.getConnection(
2. "jdbc:oracle:thin:@localhost:1521:xe","system","password");

### **3) Create the Statement object**

|  |
| --- |
| The createStatement() method of Connection interface is used to create statement. The object of statement is responsible to execute queries with the database. |

### **Syntax of createStatement() method**

**public** Statement createStatement()**throws** SQLException

1. Statement stmt=con.createStatement();

### **4) Execute the query**

|  |
| --- |
| The executeQuery() method of Statement interface is used to execute queries to the database. This method returns the object of ResultSet that can be used to get all the records of a table. |

### **Syntax of executeQuery() method**

1. **public** ResultSet executeQuery(String sql)**throws** SQLException
2. ResultSet rs=stmt.executeQuery("select \* from emp");
4. **while**(rs.next()){
5. System.out.println(rs.getInt(1)+" "+rs.getString(2));
6. }

### **5) Close the connection object**

|  |
| --- |
| By closing connection object statement and ResultSet will be closed automatically. The close() method of Connection interface is used to close the connection. |

### **Syntax of close() method**

1. **public** **void** close()**throws** SQLException

con.close();

## Step 1. Import JDBC packages

This is for making the JDBC API classes immediately available to the application program. The following import statement should be included in the program irrespective of the JDBC driver being used:

import java.sql.\*;

For individual JDBC API classes imports:

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

Connection java.sql.DriverManager.getConnection(String url) throws SQLException

Connection java.sql.DriverManager.getConnection(String url, String username, String password) throws SQLException

Connection java.sql.DriverManager.getConnection(String url, Properties info) throws SQLException

try (Connection connection = DriverManager

.getConnection("jdbc:mysql://localhost:3306/mysql\_database?useSSL=false", "root", "root");

// Step 3:Create a statement using connection object

Statement stmt = connection.createStatement();

// Step 4: Execute the query or update query

ResultSet rs = stmt.executeQuery(QUERY));

// Step 4: Process the ResultSet object.

while (rs.next()) {

int id = rs.getInt("id");

String name = rs.getString("name");

String email = rs.getString("email");

String country = rs.getString("country");

String password = rs.getString("password");

System.out.println(id + "," + name + "," + email + "," + country + "," + password);

}

## **JDBC DriverManager**

JDBC DriverManager defines objects which connect Java Applications to a JDBC driver. It is a basic service used to manage different types of JDBC database drivers.

DriverManager class have three methods getConnection(String url, String user, String password):

getConnection(String url):

**import java.sql.DriverManager;**

## **Code Example**

String driverName = "oracle.jdbc.driver.OracleDriver";

String hostName = "localhost";

String portNumber = "1521";

String serviceName = "orclpdb.localdomain";

String username="scott";

String password="tiger";

String url = "jdbc:oracle:thin:@//" + hostName + ":" +

 portNumber + "/" + serviceName ;

try {

 // Load the Oracle JDBC driver

 Class.forName(driverName);

Connection connection =

DriverManager.getConnection(url, username, password);

 } catch (Exception exp) {

 System.out.println("Exception : "

 +exp.getMessage());

 }

## **Steps**

There are several configuration steps that need to be done for the JDBC program to work.

* JDK should be installed
* Java Code editor – Eclipse IDE/IntelliJ/NetBeans is required for seamless Java development. The tutorial uses Eclipse IDE.
* Oracle Database 12g 18c..latter XE
* JDBC Driver Configuration in the IDE.

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

import java.sql.Statement;

public class JDBCExample {

   static final String DB\_URL = "jdbc:mysql://localhost/";

   static final String USER = "guest";

   static final String PASS = "guest123";

   public static void main(String[] args) {

      // Open a connection

      try(Connection conn = DriverManager.getConnection(DB\_URL, USER, PASS);

         Statement stmt = conn.createStatement();

      ) {

         String sql = "CREATE DATABASE STUDENTS";

         stmt.executeUpdate(sql);

         System.out.println("Database created successfully...");

      } catch (SQLException e) {

         e.printStackTrace();

      }

   }

}

## Technologies used

1. JDK - 1.8 /16 or later
2. MySQL - 5.7.12 MySQL connectors / oracle12g ,19c- ojdbc6.jar/ ojdbc8.jar/ ojdbc14.jar
3. IDE - Eclipse Neon
4. JDBC API - 4.2

create table users(

id int(3) primary key,

name varchar(20),

email varchar(20),

country varchar(20),

password varchar(20)

);

update

Statement statement = connection.createStatement();

// Step 3: Execute the query or update query

int result = statement.executeUpdate(UPDATE\_USERS\_SQL);

System.out.println("Number of records affected :: " + result);

Delete

// Step 2:Create a statement using connection object

Statement statement = connection.createStatement();) {

// Step 3: Execute the query or update query

int result = statement.executeUpdate(DELETE\_USERS\_SQL);

System.out.println("Number of records affected :: " + result);

# Types of Statements in JDBC

* Create Statement
* Prepared Statement
* Callable Statement

**1.**  **Create a Statement:**From the connection interface, you can create the object for this interface. It is generally used for general**–**purpose access to databases and is useful while using static SQL statements at runtime.

* **Syntax:**
* Statement statement = connection.createStatement();

***boolean execute(String SQL):***If the ResultSet object is retrieved, then it returns true else false is returned. Is used to execute [SQL DDL](https://www.geeksforgeeks.org/sql-ddl-dql-dml-dcl-tcl-commands/) statements or for dynamic SQL.

* **int executeUpdate(String SQL):** Returns number of rows that are affected by the execution of the statement, used when you need a number for INSERT, DELETE or UPDATE statements.DML
* ***ResultSet executeQuery(String SQL):***Returns a ResultSet object. Used similarly as SELECT is used in SQL DQL/DRL

// Java Program to Illustrate Setting Up of JDBC

// Importing SQL database

import java.sql.\*;

// Main class to illustrate demo of JDBC

class GFG {

    // Main driver method

    public static void main(String[] args) throws Exception

    {

        // Loading and registering drivers

        // Optional from JDBC version 4.0

        Class.forName("oracle.jdbc.OracleDriver");

        // Step 2:Establishing a connection

        Connection con = DriverManager(

            "jdbc:oracle:thin:@localhost:1521:XE",

            "username", "password");

        // Step 3: Creating statement

        Statement st = con.createStatement();

        // Step 4: Executing the query and storing the

        // result

        ResultSet rs = st.executeQuery(

            "select \* from Students where Marks >= 70");

        // Step 5: Processing the results

        while (rs.next()) {

            System.out.println(rs.getString("students"));

            System.out.println(rs.getInt("marks"));

        }

        // Step 6: Closing the connections

        // using close() method to release memory resources

        con.close();

      // Display message for successful execution of program

      System.out.println("Steps in Setting Up of JDBC");

    }

}

1. **Driver class:**The driver class for the oracle database is **oracle.jdbc.driver.OracleDriver**.
2. **Connection URL:**The connection URL for the oracle10G database is **jdbc:oracle:thin:@localhost:1521:xe** where jdbc is the API, oracle is the database, thin is the driver, localhost is the server name on which oracle is running, we may also use IP address, 1521 is the port number and XE is the Oracle service name. You may get all these information from the tnsnames.ora file.
3. **Username:**The default username for the oracle database is **system**.
4. **Password:**It is the password given by the user at the time of installing the oracle database.

|  |
| --- |
| Create a Table Before establishing connection, let's first create a table in oracle database. Following is the SQL query to create a table. |

1. create table emp(id number(10),name varchar2(40),age number(3));

import java.sql.\*;

class OracleCon{

public static void main(String args[]){

try{

//step1 load the driver class

Class.forName("oracle.jdbc.driver.OracleDriver");

//step2 create  the connection object

Connection con=DriverManager.getConnection(

"jdbc:oracle:thin:@localhost:1521:xe","system","oracle");

//step3 create the statement object

Statement stmt=con.createStatement();

//step4 execute query

ResultSet rs=stmt.executeQuery("select \* from emp");

while(rs.next())

System.out.println(rs.getInt(1)+"  "+rs.getString(2)+"  "+rs.getString(3));

//step5 close the connection object

con.close();

}catch(Exception e){ System.out.println(e);}

}

import java.sql.\*;

class FetchRecord{

public static void main(String args[])throws Exception{

Class.forName("oracle.jdbc.driver.OracleDriver");

Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","oracle");

Statement stmt=con.createStatement();

//stmt.executeUpdate("insert into emp765 values(33,'Irfan',50000)");

//int result=stmt.executeUpdate("update emp765 set name='Vimal',salary=10000 where id=33");

int result=stmt.executeUpdate("delete from emp765 where id=33");

System.out.println(result+" records affected");

con.close();

}}

# PreparedStatement interface

The PreparedStatement interface is a subinterface of Statement. It is used to execute parameterized query.

Let's see the example of parameterized query:

1. String sql="insert into emp values(?,?,?)";

As you can see, we are passing parameter (?) for the values. Its value will be set by calling the setter methods of PreparedStatement.

### **Why use PreparedStatement?**

**Improves performance**: The performance of the application will be faster if you use PreparedStatement interface because query is compiled only once.

1. **public** PreparedStatement prepareStatement(String query)**throws** SQLException{}

### **Example of PreparedStatement interface that inserts the record**

First of all create table as given below:

1. create table emp(id number(10),name varchar2(50));
2. import java.sql.\*;
3. class InsertPrepared{
4. public static void main(String args[]){
5. try{
6. Class.forName("oracle.jdbc.driver.OracleDriver");
8. Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","oracle");
10. PreparedStatement stmt=con.prepareStatement("insert into Emp values(?,?)");
11. stmt.setInt(1,101);//1 specifies the first parameter in the query
12. stmt.setString(2,"Ratan");
14. int i=stmt.executeUpdate();
15. System.out.println(i+" records inserted");
17. con.close();
19. }catch(Exception e){ System.out.println(e);}
21. }
22. }

### **Example of PreparedStatement interface that updates the record**

PreparedStatement stmt=con.prepareStatement("update emp set name=? where id=?");

stmt.setString(1,"Sonoo");//1 specifies the first parameter in the query i.e. name

stmt.setInt(2,101);

int i=stmt.executeUpdate();

System.out.println(i+" records updated");

### **Example of PreparedStatement interface that deletes the record**

PreparedStatement stmt=con.prepareStatement("delete from emp where id=?");

stmt.setInt(1,101);

int i=stmt.executeUpdate();

System.out.println(i+" records deleted");

### **Example of PreparedStatement interface that retrieve the records of a table**

PreparedStatement stmt=con.prepareStatement("select \* from emp");

ResultSet rs=stmt.executeQuery();

while(rs.next()){

System.out.println(rs.getInt(1)+" "+rs.getString(2));

}

[**Statement interface**](http://www.javaguides.net/2018/10/jdbc-statement-interface.html) *executeUpdate()* method executes the given SQL statement, which may be an INSERT, UPDATE, or DELETE statement or an SQL statement that returns nothing, such as an SQL DDL statement.

// Step 2:Create a statement using connection object

Statement statement = connection.createStatement();) {

// Step 3: Execute the query or update query

int result = statement.executeUpdate(DELETE\_USERS\_SQL);

System.out.println("Number of records affected :: " + result);

// Step 2:Create a statement using connection object

PreparedStatement preparedStatement = connection.prepareStatement(INSERT\_USERS\_SQL)) {

preparedStatement.setInt(1, 1);

preparedStatement.setString(2, "Tony");

preparedStatement.setString(3, "tony@gmail.com");

preparedStatement.setString(4, "US");

preparedStatement.setString(5, "secret");

System.out.println(preparedStatement);

// Step 3: Execute the query or update query

preparedStatement.executeUpdate();

} catch (SQLException e) {

// print SQL exception information

printSQLException(e);

}

// Step 4: try-with-resource statement will auto close the connection.

}

public static void printSQLException(SQLException ex) {

for (Throwable e: ex) {

if (e instanceof SQLException) {

e.printStackTrace(System.err);

System.err.println("SQLState: " + ((SQLException) e).getSQLState());

System.err.println("Error Code: " + ((SQLException) e).getErrorCode());

System.err.println("Message: " + e.getMessage());

Throwable t = ex.getCause();

while (t != null) {

System.out.println("Cause: " + t);

t = t.getCause();

}

}

}

}

}

PreparedStatement pstmt = con.prepareStatement("UPDATE EMPLOYEES SET SALARY = ? WHERE ID = ?");

pstmt.setBigDecimal(1, 153833.00)

pstmt.setInt(2, 110592)

// Step 2:Create a statement using connection object

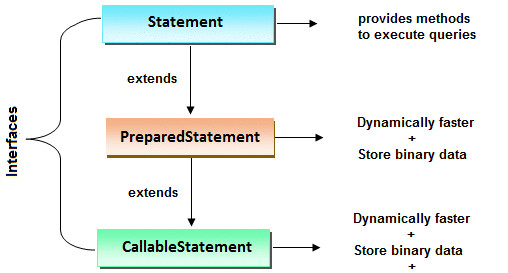
PreparedStatement preparedStatement = connection.prepareStatement(UPDATE\_USERS\_SQL)) {

preparedStatement.setString(1, "Ram");

preparedStatement.setInt(2, 1);

// Step 3: Execute the query or update query

preparedStatement.executeUpdate();

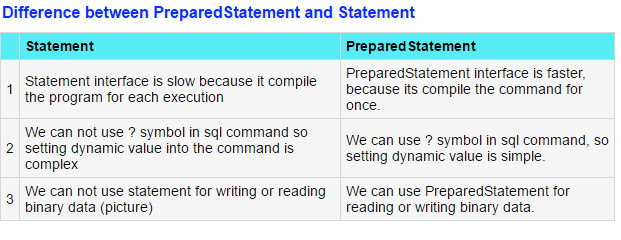


The JDBC **Statement**, **CallableStatement**, and **PreparedStatement**interfaces define the methods and properties that enable you to send SQL or PL/SQL commands and receive data from your database. when the connection has been established to the database.

The **Statement** is used for executing a static SQL statement.

The **PreparedStatement** is used for executing a precompiled SQL statement.

The **CallableStatement** is an interface which is used to execute SQL stored procedures, cursors, and Functions.



##### **CRUD Operations in Java using JDBC:**

CRUD is the acronym for the following four operations.

1. **C- INSERTION**
2. **R- RETRIEVAL**
3. **U- UPDATION**
4. **D- DELETION**

**CREATE TABLE ACCOUNT (accnonumber(8) primary key, name varchar2(12), balance number(8,2));**

import java.sql.\*;

class AccountStoringApplication

{

    public static void main(String[] args) throws ClassNotFoundException, SQLException

    {

     Class.forName("oracle.jdbc.driver.OracleDriver");

     Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","System", "pranaya");

     Statement st = con.createStatement();

        int c = st.executeUpdate("insert into account values(1005, 'pranaya', 2345)");

        System.out.println(c + "account stored successfully");

        int c = st.executeUpdate("insert into account values(1006, 'kumar', 5345)");

        System.out.println(c + "" more account stored successfully");

        st.close();

        con.close();

    }

}

##### **Update Operations using JDBC in Java:**

The following JDBC APPLICATION is used to update the accounts by adding Rs 2000 to each account in the Account table. The SQL statement for updating is: **update account set balance = balance+2000**

import java.sql.\*;

class UpdateAccountApplication

{

    public static void main (String[]args) throws ClassNotFoundException, SQLException

    {

        Class.forName ("oracle.jdbc.driver.OracleDriver");

        Connection con = DriverManager.getConnection ("jdbc:oracle:thin:@localhost:1521:xe", "System", "pranaya");

        Statement st = con.createStatement ();

        int rows = st.executeUpdate ("update account set balance = balance+2000");

        System.out.println (rows + " rows modified");

        st.close ();

        con.close ();

    }

}

import java.sql.\*;

import java.util.\*;

class AccountDetails

{

    public static void main (String[]args) throws ClassNotFoundException, SQLException

    {

        Scanner sc = new Scanner (System.in);

        System.out.println ("ENTER ACCOUNT NUMBER");

        int ano = sc.nextInt ();

        Class.forName ("oracle.jdbc.driver.OracleDriver");

        Connection con = DriverManager.getConnection ("jdbc:oracle:thin:@localhost:1521:xe", "System", "pranaya");

        Statement st = con.createStatement ();

        ResultSet rs = st.executeQuery ("select \* from account where accno =" + ano);

        if (rs.next ())

        {

            System.out.println ("account no:     " + rs.getInt (1));

            System.out.println ("acc holder name:" + rs.getString (2));

            System.out.println ("balance :       " + rs.getFloat (3));

            System.out.println ("address:        " + rs.getString (4));

        }

        else

            System.out.println ("account doesnot exist");

        rs.close ();

        st.close ();

        con.close ();

    }

}

**ResultSet rs = st.executeQuery(“select \* from account”);**

while(rs.next())

{

      System.out.println(rs.getInt(1));

      System.out.println(rs.getString(2));

      System.out.println(rs.getFloat(3));

}

##### **What are the methods of ResultSet that deal with the Cursor?**

1. next()
2. previous()
3. first()
4. last()
5. afterLast()
6. beforeFirst()
7. absolute()
8. relative()
9. moveToInsertRow()
10. moveToCurrentRow()

**Note:** these 10 methods move the cursor.

1. int getRow()
2. boolean isFirst()
3. boolean isLast()
4. boolean isAfterLast()
5. Boolean isBeforeFirst()

import java.sql.\*;

class AccountStoringApplication

{

    public static void main (String[]args) throws ClassNotFoundException, SQLException

    {

        Class.forName ("oracle.jdbc.driver.OracleDriver");

        Connection con = DriverManager.getConnection ("jdbc:oracle:thin:@localhost:1521:xe", "System", "pranaya");

        PreparedStatement ps = con.prepareStatement ("INSERT INTO ACCOUNT VALUES(?,?,?,?)");

        ps.setInt (1, 1001);

        ps.setString (2, "ANURAG");

        ps.setFloat (3, 2345);

        ps.setString (4, "puri");

        int c = ps.executeUpdate ();

        System.out.println (c + "account created successfully");

        ps.setInt (1, 1002);

        ps.setString (2, "PRANAYA");

        ps.setFloat (3, 2345);

        ps.setString (4, "JAJPUR");

        c = ps.executeUpdate ();

        System.out.println (c + " MORE account created successfully");

        ps.close ();

        con.close ();

    }

}

##### **Update Operation using Prepared Statement:**

SQL Statement: **UPDATE ACCOUNT SET BALANCE = BALANCE + 500 WHERE ACCNO = 1001;**

Instead of writing this SQL statement, write this SQL statement with place holder (?) as shown in the below program.

import java.sql.\*;

class AccountUpdatingApplication

{

    public static void main (String[]args) throws ClassNotFoundException, SQLException

    {

        Class.forName ("oracle.jdbc.driver.OracleDriver");

        Connection con = DriverManager.getConnection ("jdbc:oracle:thin:@localhost:1521:xe", "System", "pranaya");

        PrepaaredStatement ps = con.prepareStatement("UPDATE ACCOUNT SET BALANCE = BALANCE + ? WHERE ACCNO = ?");

        ps.setFloat(1,500);

        ps.setInt(2,1001);

        ps.executeUpdate();

        ps.setFloat(1,500);

        ps.setInt(2,1002);

        ps.executeUpdate();

        ps.close ();

        con.close ();

    }

}

import java.sql.\*;

class AccountUpdatingApplication

{

    public static void main (String[]args) throws ClassNotFoundException, SQLException

    {

        Class.forName ("oracle.jdbc.driver.OracleDriver");

        Connection con = DriverManager.getConnection ("jdbc:oracle:thin:@localhost:1521:xe", "System", "pranaya");

        PrepaaredStatement ps = con.prepareStatement("UPDATE ACCOUNT SET BALANCE = BALANCE + ? WHERE ACCNO = ?");

        ps.setFloat(1,500);

        ps.setInt(2,1001);

        ps.executeUpdate();

        ps.setFloat(1,500);

        ps.setInt(2,1002);

        ps.executeUpdate();

        ps.close ();

        con.close ();

    }

}

##### **Delete Operation using Prepared Statement in Java:**

SQL Statement:**DELETE FROM ACCOUNT WHERE ACCNO = 1001;**

Instead of writing this SQL statement, write this SQL statement with a placeholder (?) as shown in the below program.

import java.sql.\*;

class AccountClosingApplication

{

    public static void main (String[]args) throws ClassNotFoundException, SQLException

    {

        Class.forName ("oracle.jdbc.driver.OracleDriver");

        Connection con = DriverManager.getConnection ("jdbc:oracle:thin:@localhost:1521:xe", "System", "pranaya");

        PrepaaredStatement ps = con.prepareStatement("DELETE FROM ACCOUNT WHERE ACCNO = ?");

        ps.setInt(1,1001);

        ps.executeUpdate();

        ps.setInt(2,1001);

        ps.executeUpdate();

        ps.close ();

        con.close ();

    }

}

##### **Retrieve Operation using Prepared Statement in Java:**

SQL Statement: **SELECT BALANCE FROM ACCOUNT WHERE ACCNO = 1001;**

Instead of writing this SQL statement, write this SQL statement with a placeholder (?) as shown in the below program.

import java.sql.\*;

import java.util.\*;

class AccountDetailsInformation

{

    public static void main (String[]args) throws ClassNotFoundException, SQLException

    {

        Class.forName ("oracle.jdbc.driver.OracleDriver");

        Connection con = DriverManager.getConnection ("jdbc:oracle:thin:@localhost:1521:xe", "System", "pranaya");

        PreparedStatement ps = con.prepareStatement ("select balance from account where accno = ?");

        ps.setInt (1, 1001);

        ResultSet rs = ps.executeQuery ();

        if (rs.next ())

        {

            System.out.println ("A/C balance is Rs." + rs.getFloat (1));

        }

        else

            System.out.println ("A/C doesn't Exist");

        rs.close ();

        ps.setInt (1, 1004);

        ResultSet rs1 = ps.executeQuery ();

        if (rs1.next ())

        {

            System.out.println ("A/C balance is Rs." + rs1.getFloat (1));

        }

        else

            System.out.println ("A/C doesn't Exist");

        rs1.close ();

        ps.close ();

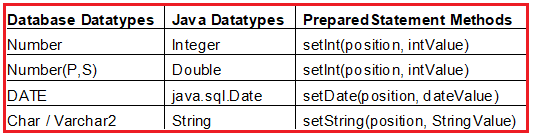
        con.close ();

    }

}

##### **What are the Positional Parameters?**

When we use PreparedStatement it should contain the query with positional parameters. Positional Parameters are specified using the “?” symbol. Positional Parameters are used to supply the values to the query. To supply the values for positional parameters we have to use the following setter methods:



### 1. What is JDBC?

JDBC stands for Java Database Connectivity, a standard Java API for database-independent connectivity between the Java programming language and a wide range of databases.

### 2. What is a connection?

A connection is an Interface that includes all methods for contacting a database. The connection object is a source of communication context, i.e., all communication with the database with the help of the connection object only.

### 3. What is a statement?

Statement abstracts an SQL statement that must be parsed, planned, compiled, passed and executed.

### 4. What is the use of a ResultSet?

ResultSet is the object that holds the data retrieved from a database after executing an SQL query with the use of Statement objects. ResultSet helps move through the data freely. The java.sql.ResultSet interface represents the result set of a database query.

### 5. What is a JDBC Driver?

JDBC driver is an interface that enables the interaction of Java applications with a database. Drivers are required for each database when JDBC connects with individual databases. The JDBC driver acts as a connection to the database, implementing the necessary protocols for transferring the query and result between client and database.

### 6. What is a JDBC DriverManager?

The class that handles a list of database drivers is called JDBC DriverManager. The connection requests are matched from the java application to the database driver using communication subprotocol.

### 7. What is a RowSet?

A RowSet is an object that holds tabular data in an easily adaptable way to use rather than a result set. JavaBeans components are the RowSet objects.

### 8. Describe a general JDBC Architecture

The architecture of JDBC consists of two layers JDBC Driver API (The layer supports JDBC Manager-to-Driver Connection) and JDBC API (This layer provides the application-to-JDBC Manager connection).

### 9. What are the components that are common in JDBC API?

The components and interfaces included in JDBC API are Driver, Connection, DriverManager, Statement, SQLException ResultSet.

### 10. Which type of JDBC driver is the fastest one?

JDBC Net pure Java driver(Type 4) is the fastest driver of all as it converts the JDBC calls into vendor-specific protocol calls and interacts directly with the database.

### 11. What are the different types of JDBC Statements?

Statements used in JDBC are

**Statement** − which is a regular SQL statement.  
**PreparedStatement** − is more efficient than a SQL Statement.  
**CallableStatement**− is used to call stored procedures in the database.

### 12. How does JDBC handle the data types of Java and database?

The JDBC driver converts the Java data type to the appropriate JDBC type. It uses default mapping for most data types. For example, a Java int is converted to an SQL INTEGER.

### 13. How is the fastness of a JDBC driver measured?

The size of the driver code, Quality of the driver code, database server and its load, Network topology, Number of times the request is translated to a different API are all the number of issues that decide the speed of a JDBC driver.

### 14. How can you view a result set?

To view the result set, we use the get methods. Each get method in the  ResultSet interface provides two versions.

* One accepts the column name.
* One accepts the column index.

Ex: getInt(String columnName), getInt(int columnIndex)

### 15. What does set AutoCommit do?

AutoCommit ensures that every individual SQL statement is committed right after its execution when it is set in auto-commit mode. When created, any connection is in auto-commit mode by default. When the auto-commit is false, no SQL statements will be committed until explicitly called by the commit method.

### 1) What is the JDBC?

JDBC stands for Java Database Connectivity. JDBC is a Java API that communicates with the database and execute SQLquery.

### 2) What is a JDBC driver and how many JDBC drivers are available?

JDBC driver contains classes and interfaces that help Java application and database.

There are 4 types of JDBC drivers.

1. Type 1 driver or JDBC-ODBC bridge driver.
2. Type 2 driver or Native-[API](https://career.guru99.com/top-20-questions-on-api-testing/), partly Java driver.
3. Type 3 driver or Network Protocol, pure Java driver.
4. Type 4 driver or Native-protocol, pure Java driver.

### 4) Which JDBC driver is the fastest driver?

Type 4 driver or Native-protocol, pure Java driver, is the fastest driver.

### 5) What are the JDBC API components?

There are four types of components

1. JDBC API
2. JDBC Driver Manager
3. JDBC Test Suite
4. JDBC-ODBC Bridge

### 6) What are the JDBC statements?

There are 3 types of JDBC Statements, as given below:

1. **Statement**: It will execute [SQL](https://www.guru99.com/sql-server-questions.html) query (static SQL query) against the database.
2. **Prepared Statement:** Used when we want to execute SQL statement repeatedly.  Input data is dynamic and  taken input at the run time.
3. **Callable Statement:** Used when we want to execute stored procedures.

### 20) How many packages are available in JDBC API?

Two types of packages are available in JDBC API

1. java.sql
2. javax.sql

### 9. What are the differences between ODBC and JDBC?

| **ODBC(Open Database Connectivity)** | **JDBC(Java Database Connectivity)** |
| --- | --- |
| ODBC can be used for languages like C, C++, Java, etc. | JDBC is used only for the Java language |
| We can use ODBC only for the Windows platform, thus it is platform-dependent. | We can use JDBC on any platform, thus it is platform-independent |
| Most of the ODBC Drivers developed in native languages like C, C++ | JDBC drivers are developed using the Java language |
| It is not recommended to use ODBC for Java applications, because of low performance due to internal conversion. | It is highly recommended to use JDBC for Java applications because there are no performance issues. |
| ODBC is procedural. | JDBC is Object Oriented. |

**ap from Scaler**

[**Show My Career Plan**](https://www.scaler.com/career-plan?utm_source=ib&utm_medium=top_nudge&utm_campaign=CareerPlan&utm_content=)

[Practice](https://www.interviewbit.com/practice/)

Resources

[Contests](https://www.interviewbit.com/contests/)

Online IDE

Free Mock

[EventsNew](https://www.interviewbit.com/events/)[Scaler](https://www.scaler.com/?utm_source=ib&utm_campaign=ib_menu&utm_medium=nav_menu)

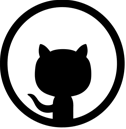


[Sign in](javascript:void(0))

[](https://www.interviewbit.com/users/auth/google_oauth2/)

[](https://www.interviewbit.com/users/auth/linkedin/)

[](https://www.interviewbit.com/users/auth/facebook/)

[](https://www.interviewbit.com/users/auth/github/)

# JDBC Interview Questions

Last updated on Jan 02, 2023

[**Download PDF**](javascript:void(0))

Share with:



Copied

* [**JDBC Interview Questions for Freshers**](https://www.interviewbit.com/jdbc-interview-questions/#jdbc-interview-questions-for-freshers)
  + [1.What is JDBC in Java?](https://www.interviewbit.com/jdbc-interview-questions/#java-jdbc)
  + [2.What is ResultSet?](https://www.interviewbit.com/jdbc-interview-questions/#jdbc-resultset)
  + [3.What is JDBC driver?](https://www.interviewbit.com/jdbc-interview-questions/#jdbc-driver)
  + [4.What is DriverManager in JDBC?](https://www.interviewbit.com/jdbc-interview-questions/#jdbc-drivermanager)
  + [5.Which JDBC driver is fastest and used more commonly?](https://www.interviewbit.com/jdbc-interview-questions/#fastest-jdbc-driver)
  + [6.Which data types are used for storing the image and file in the database table?](https://www.interviewbit.com/jdbc-interview-questions/#data-types-used-for-storing-images-and-files-in-database-table)
  + [7.What is stored procedure? What are the parameter types in stored procedure?](https://www.interviewbit.com/jdbc-interview-questions/#explain-stored-procedure-and-its-parameter-types)
  + [8.What do you mean by DatabaseMetaData and why we are using it?](https://www.interviewbit.com/jdbc-interview-questions/#what-is-jdbc-database-metadata-and-its-uses)
  + [9.What are the differences between ODBC and JDBC?](https://www.interviewbit.com/jdbc-interview-questions/#odbc-vs-jdbc)
  + [10.What is Rowset?](https://www.interviewbit.com/jdbc-interview-questions/#jdbc-rowset)
* [**JDBC Interview Questions for Experienced**](https://www.interviewbit.com/jdbc-interview-questions/#jdbc-interview-questions-for-experienced)
  + [11.What are the different types of JDBC drivers in Java? Explain each with an example.](https://www.interviewbit.com/jdbc-interview-questions/#explain-each-type-of-jdbc-drivers-in-java-with-an-example)
  + [12.What are difference between ResultSet and RowSet?](https://www.interviewbit.com/jdbc-interview-questions/#resultset-vs-rowset)
  + [13.Explain the types of ResultSet.](https://www.interviewbit.com/jdbc-interview-questions/#jdbc-resultset-types)
  + [14.Explain JDBC API components.](https://www.interviewbit.com/jdbc-interview-questions/#jdbc-api-components)
  + [15.What are the types of JDBC statements?](https://www.interviewbit.com/jdbc-interview-questions/#jdbc-statement-types)
  + [16.Explain JDBC Batch processing.](https://www.interviewbit.com/jdbc-interview-questions/#jdbc-batch-processing)
  + [17.What is the difference between Statement and PreparedStatement?](https://www.interviewbit.com/jdbc-interview-questions/#statement-vs-preparedstatement)
  + [18.What is DataSource in JDBC? What are its benefits?](https://www.interviewbit.com/jdbc-interview-questions/#explain-jdbc-datasource-and-list-its-benefits)
  + [19.Explain the difference between execute(), executeQuery() and executeUpdate() methods in JDBC.](https://www.interviewbit.com/jdbc-interview-questions/#jdbc-executequery-vs-executeupdate-vs-execute-method)
  + [20.Explain the types of RowSet available in JDBC.](https://www.interviewbit.com/jdbc-interview-questions/#jdbc-rowset-types)
  + [21.Explain the usage of the getter and setter methods in ResultSet.](https://www.interviewbit.com/jdbc-interview-questions/#explain-the-usage-of-resultset-getter-and-setter-methods)
  + [22.What is meant by a locking system in JDBC?](https://www.interviewbit.com/jdbc-interview-questions/#locking-system-in-jdbc)
  + [23.What is database connection pooling? What are the advantages of connection pool?](https://www.interviewbit.com/jdbc-interview-questions/#benefits-of-database-connection-pooling)
  + [24.What is “Dirty read” in terms of database?](https://www.interviewbit.com/jdbc-interview-questions/#dirty-read-in-database)
  + [25.What causes “No suitable driver” error?](https://www.interviewbit.com/jdbc-interview-questions/#what-causes-no-suitable-driver-error)
  + [26.What is JDBC Connection? Explain steps to get JDBC database connection in a simple Java program.](https://www.interviewbit.com/jdbc-interview-questions/#explain-the-steps-to-setup-a-jdbc-database-connection-in-java)
  + [27.How to use JDBC API to call Stored procedures?](https://www.interviewbit.com/jdbc-interview-questions/#how-to-use-jdbc-api-to-call-stored-procedures)
  + [28.What are the types of JDBC architecture?](https://www.interviewbit.com/jdbc-interview-questions/#jdbc-architecture-types)
  + [29.What is JDBC Transaction Management and why is it needed?](https://www.interviewbit.com/jdbc-interview-questions/#what-is-jdbc-transaction-management-and-why-is-it-needed)
  + [30.Explain the benefits of PreparedStatement over Statement.](https://www.interviewbit.com/jdbc-interview-questions/#explain-the-benefits-of-preparedstatement-over-statement)
  + [31.Explain the methods available for Transaction Management in JDBC.](https://www.interviewbit.com/jdbc-interview-questions/#transaction-management-in-jdbc)
  + [32.Give few examples of most common exceptions in JDBC.](https://www.interviewbit.com/jdbc-interview-questions/#jdbc-exceptions)
  + [33.What is Two phase commit in JDBC?](https://www.interviewbit.com/jdbc-interview-questions/#what-is-two-phase-commit-in-jdbc)
  + [34.What are the isolation levels of connections in JDBC?](https://www.interviewbit.com/jdbc-interview-questions/#jdbc-isolation-levels-of-connections)
  + [35.How to create a table dynamically from a JDBC application?](https://www.interviewbit.com/jdbc-interview-questions/#how-to-create-a-table-dynamically-from-a-jdbc-application)
  + [36.Conclusion](https://www.interviewbit.com/jdbc-interview-questions/#conclusion)
* [**JDBC MCQ**](https://www.interviewbit.com/jdbc-interview-questions/#jdbc-mcq)

### Introduction to JDBC:

JDBC is an Application Programming Interface(API) for Java, which is helpful for interaction with the database and for executing the SQL query. JDBC is an abbreviation used for Java Database Connectivity. It uses JDBC drivers for connecting with the [database](https://www.interviewbit.com/dbms-interview-questions/). JDBC API is used to access tabular data stored in relational databases like Oracle, MySQL, MS Access, etc.

### Components of JDBC:

There are four major components of JDBC using which it can interact with a database. They are:

1. **JDBC API**: It provides different methods and interfaces for easier communication with the database. By using this, applications are able to execute SQL statements, retrieve results and make updation to the database. It has two packages as follows which consist of Java SE and Java EE platforms to exhibit Write Once Run Everywhere(WORA) capabilities.
   1. java.sql.\*;
   2. javax.sql.\*;  
      Also, it provides a standard for connecting a database to a client application.
2. **JDBC DriverManager**: It is the class in JDBC API. It loads the JDBC driver in a Java application for establishing a connection with the database. It is useful in making a database-specific call for processing the user request.
3. **JDBC Test suite**: It is used to test the operations like insertion, deletion, updation etc., being performed by JDBC Drivers.
4. **JDBC-ODBC bridge drivers**: It will connect database drivers to the database. JDBC-ODBC bridge interprets JDBC method call to the ODBC function call. It will use sun.jdbc.odbc package, which consists of the native library to access characteristics of ODBC.

### Scope of JDBC:

Earlier, ODBC API was used as the database API to connect with the database and execute the queries. But, ODBC API uses C language for ODBC drivers(i.e. platform-dependent and unsecured). Hence, Java has defined its own JDBC API that uses JDBC drivers, which offers a natural Java interface for communicating with the database through SQL. JDBC is required to provide a “pure Java” solution for the development of an application using [Java programming](https://www.interviewbit.com/java-interview-questions/).

**Crack your next tech interview with confidence!**

Take a free mock interview, get instant⚡️ feedback and recommendation💡

**Attempt Now**

**Events**|Powered By

* [**Software Dev**](https://www.interviewbit.com/jdbc-interview-questions/#sde_events)
* [Data Science](https://www.interviewbit.com/jdbc-interview-questions/#dsml_events)
* [All Events](https://www.interviewbit.com/jdbc-interview-questions/#all_events)

[View All](https://www.interviewbit.com/events)

## JDBC Interview Questions for Freshers

### 1. What is JDBC in Java?

JDBC(Java Database Connectivity) is a Java API, which is helpful in interaction with the database to retrieve, manipulate and process the data using SQL. It will make use of JDBC drivers for connecting with the database. By using JDBC, we can access tabular data stored in various types of relational databases such as Oracle, MySQL, MS Access, etc.

### 2. What is ResultSet?

* The java.sql.ResultSet interface represents the database result set, which is obtained after the execution of SQL query using Statement objects.
* Object of ResultSet maintains a cursor pointing to the current row of data in the result set. Initially, the cursor is located before the first row. Then the cursor is moved to the next row by using the next() method. The next() method can be used to iterate through the result set with the help of a while loop. If there are no further rows, the next() method will return false.
* Example for the creation of ResultSet is given below:  
  ResultSet rs = con.executeQuery(sqlQuery);

### 3. What is JDBC driver?

JDBC driver is a software component having various classes and interfaces, that enables the Java application to interact with a database.

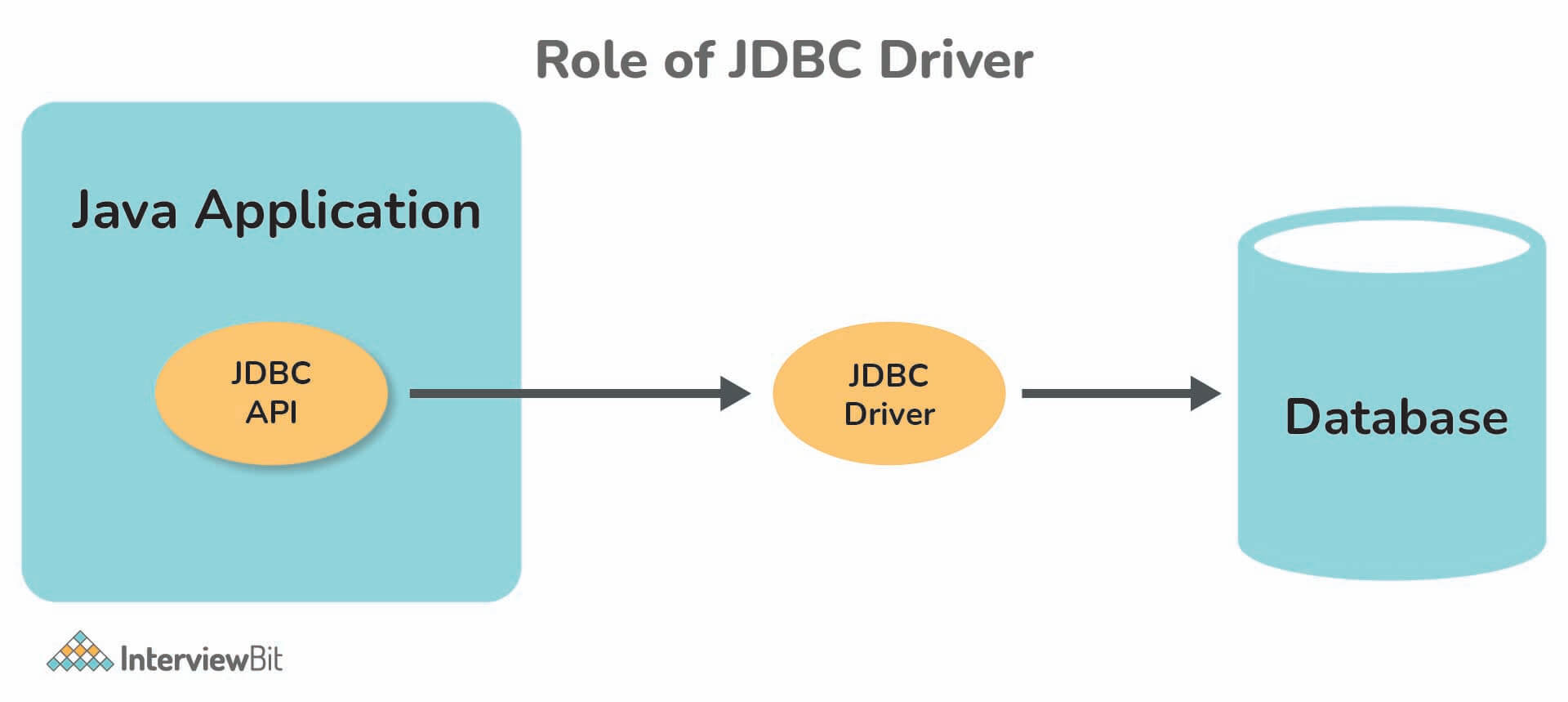
To connect with individual databases, JDBC requires particular drivers for each specific database. These drivers are provided by the database vendor in addition to the database. For example:

* MySQL Connector/J is the official JDBC driver for MySQL and we can locate the mysql-connector-java-<version>-bin.jar file among the installed files. On windows, this file can be obtained at

C:\Program Files (x86)\MySQL\MySQL Connector J\mysql-connector-java-5.1.30-bin.jar.

* JDBC driver of Oracle 10G is ojdbc14.jar and it can be obtained in the installation directory of an Oracle at …/Oracle/app/oracle/product/10.2.0/server/jdbc/lib .

JDBC driver provides the connection to the database. Also, it implements the protocol for sending the query and result between client and database.



**You can download a PDF version of Jdbc Interview Questions.**

[**Download PDF**](javascript:void(0))

### 4. What is DriverManager in JDBC?

* JDBC DriverManager is a static class in Java, through which we manage the set of JDBC drivers that are available for an application to use.
* Multiple JDBC drivers can be used concurrently by an application, if necessary. By using a Uniform Resource Locator(URL), each application specifies a JDBC driver.
* When we load the JDBC Driver class into an application, it registers itself to the DriverManager by using Class.forName() or DriverManager.registerDriver(). To check this, you can have a look into the source code of JDBC Driver classes. After this, when we call DriverManager.getConnection() method by passing the details regarding database configuration, DriverManager will make use of registered drivers to obtain the connection and return it to the caller program.

### 5. Which JDBC driver is fastest and used more commonly?

**JDBC Net pure Java driver(Type 4 driver**) is the fastest driver for localhost and remote connections because it directly interacts with the database by converting the JDBC calls into vendor-specific protocol calls.

### 6. Which data types are used for storing the image and file in the database table?

* **BLOB** data type is used to store the image in the database. We can also store videos and audio by using the BLOB data type. It stores the binary type of data.
* **CLOB**data type is used to store the file in the database. It stores the character type of data.

### 7. What is stored procedure? What are the parameter types in stored procedure?

* Stored procedure is a group of SQL queries that are executed as a single logical unit to perform a specific task. Name of the procedure should be unique since each procedure is represented by its name.
* For example, operations on an employee database like obtaining information about an employee could be coded as stored procedures that will be executed by an application. Code for creating a stored procedure named GET\_EMP\_DETAILS is given below:

DELIMITER $$

DROP PROCEDURE IF EXISTS `EMP`.`GET\_EMP\_DETAILS` $$

CREATE PROCEDURE `EMP`.`GET\_EMP\_DETAILS`

(IN EMP\_ID INT, OUT EMP\_DETAILS VARCHAR(255))

BEGIN

SELECT first INTO EMP\_DETAILS

FROM Employees

WHERE ID = EMP\_ID;

END $$

DELIMITER ;

Stored procedures are called using CallableStatement class available in JDBC API. Below given code demonstrates this:

CallableStatement cs = con.prepareCall("{call GET\_EMP\_DETAILS(?,?)}");

ResultSet rs = cs.executeQuery();

* Three types of parameters are provided in the stored procedures. They are:
  + **IN**: It is used for passing the input values to the procedure. With the help of setXXX() methods, you can bind values to IN parameters.
  + **OUT**: It is used for getting the value from the procedure. With the help of getXXX() methods, you can obtain values from OUT parameters.
  + **IN/OUT**: It is used for passing the input values and obtaining the value to/from the procedure. You bind variable values with the setXXX() methods and obtain values with the getXXX() methods.

### 8. What do you mean by DatabaseMetaData and why we are using it?

* DatabaseMetaData is an interface that provides methods to obtain information about the database.
* We can use this for getting database-related informations, such as database name, database version, driver name, the total number of tables or views, etc.

### 9. What are the differences between ODBC and JDBC?

| **ODBC(Open Database Connectivity)** | **JDBC(Java Database Connectivity)** |
| --- | --- |
| ODBC can be used for languages like C, C++, Java, etc. | JDBC is used only for the Java language |
| We can use ODBC only for the Windows platform, thus it is platform-dependent. | We can use JDBC on any platform, thus it is platform-independent |
| Most of the ODBC Drivers developed in native languages like C, C++ | JDBC drivers are developed using the Java language |
| It is not recommended to use ODBC for Java applications, because of low performance due to internal conversion. | It is highly recommended to use JDBC for Java applications because there are no performance issues. |
| ODBC is procedural. | JDBC is Object Oriented. |

### 10. What is Rowset?

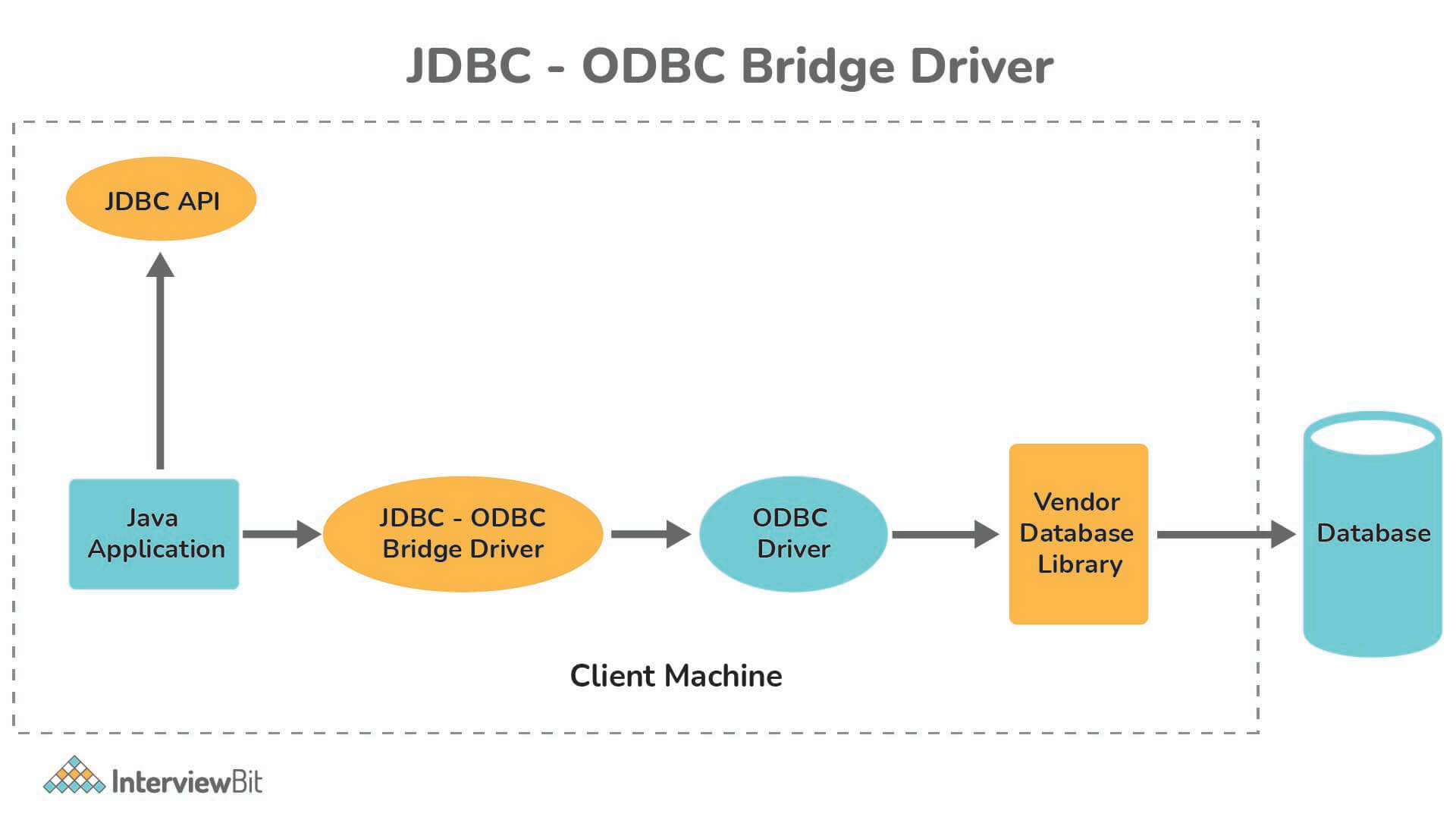
* A RowSet is an object that encapsulates a row set from either JDBC result sets or tabular data sources such as files or spreadsheets. It supports component-based development models like JavaBeans, with the help of a standard set of properties and event notifications.
* The advantages of using RowSet are:
  + It is easier and flexible to use.
  + It is Scrollable and Updatable by default.

## JDBC Interview Questions for Experienced

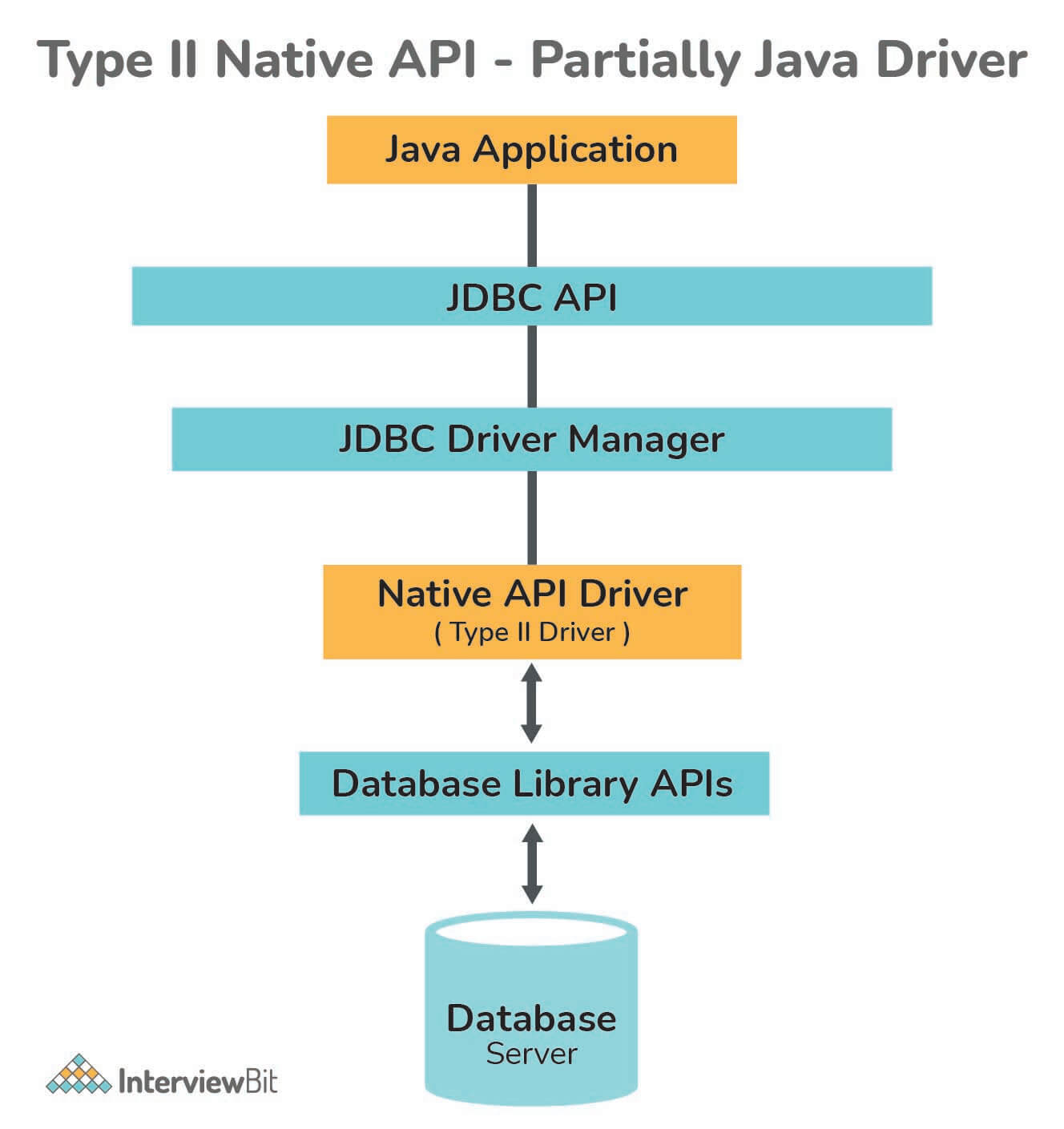
### 11. What are the different types of JDBC drivers in Java? Explain each with an example.

There are four types of JDBC drivers in Java. They are:

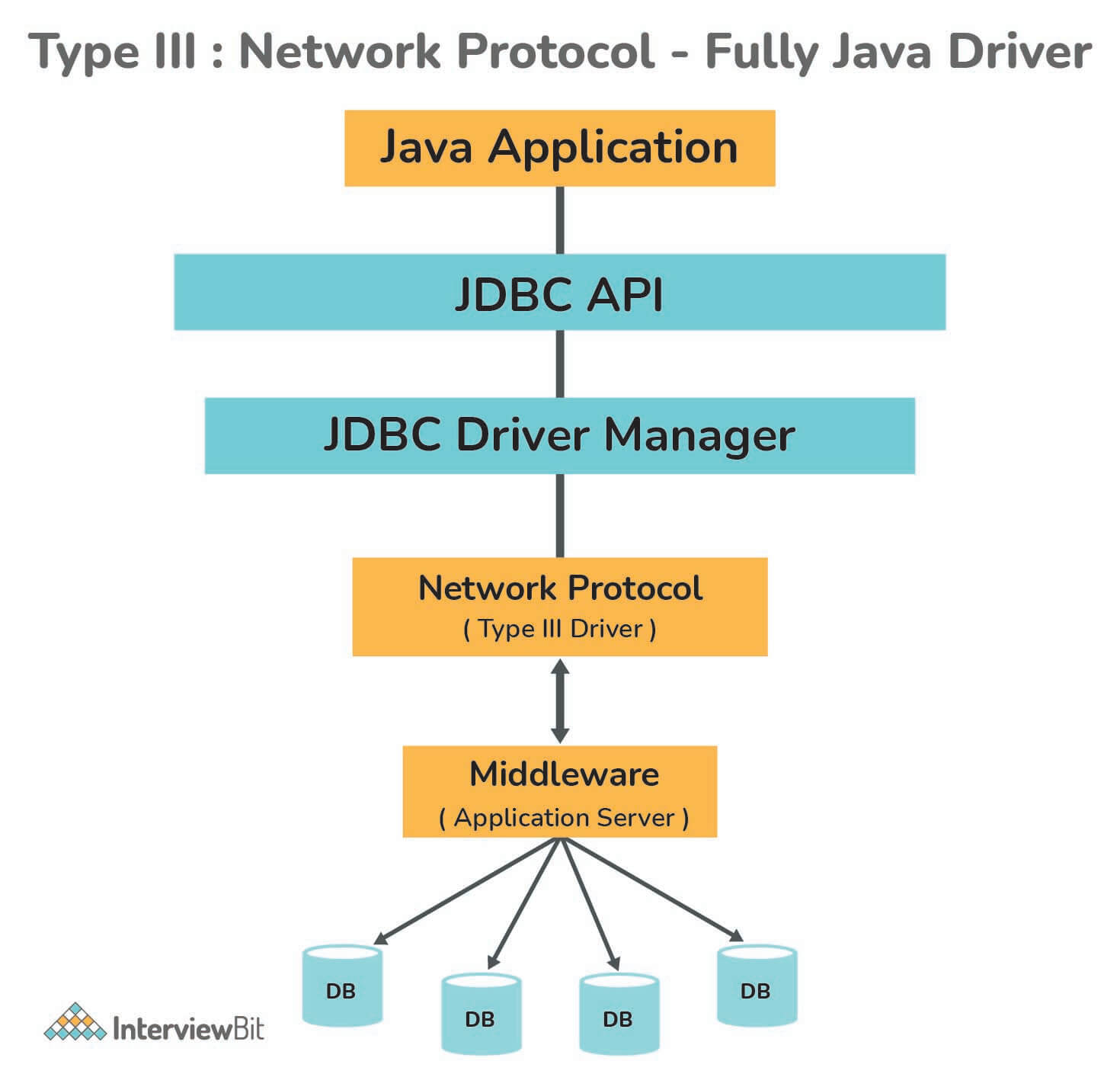
* **Type I: JDBC - ODBC bridge driver**
  + In this, the JDBC–ODBC bridge acts as an interface between the client and database server. When a user uses a Java application to send requests to the database using JDBC–ODBC bridge, it converts the JDBC API into ODBC API and then sends it to the database. When the result is received from the database, it is sent to ODBC API and then to JDBC API.
  + It is platform-dependent because it uses ODBC which depends on the native library of the operating system. In this, JDBC–ODBC driver should be installed in every client system and database must support for ODBC driver.'
  + It is easier to use but it gives low performance because it involves the conversion of JDBC method calls to the ODBC method calls.



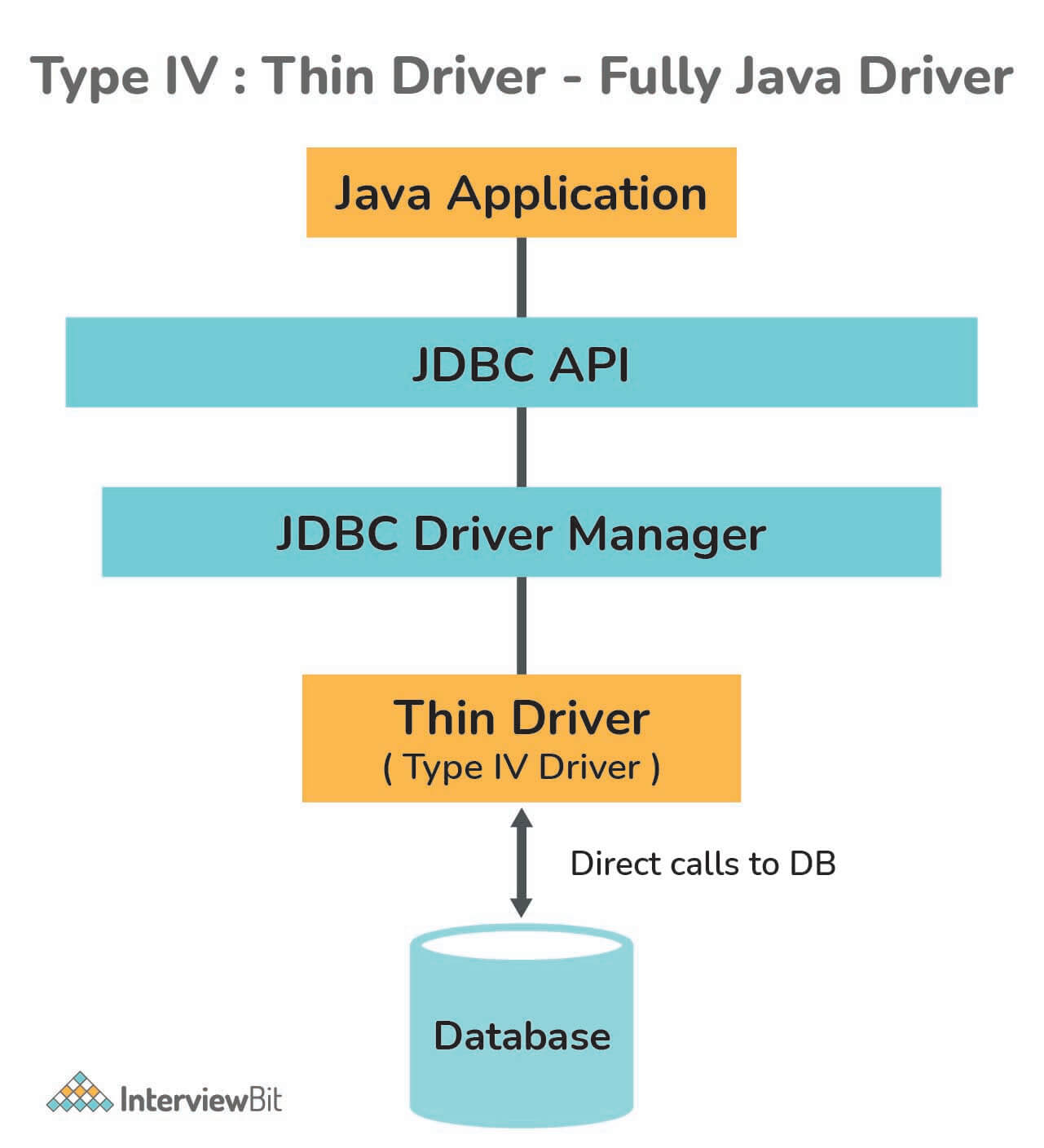
* **Type II: Native API – Partially Java Driver:**
  + It is almost similar to a Type I driver. Here, native code replaces the ODBC part. This native code part is targeted at a particular database product. It uses libraries of the client-side of the database. This Type II Driver converts the JDBC method calls to native calls of the database native API.
  + When the database gets the requests from the user, the requests are processed and sends the results back in the native format which is then converted into JDBC format and pass it to the Java application.
  + It was instantly adopted by the database vendors because it was quick and cheaper to implement. This driver gives faster response and performance compared to the Type I driver.



* **Type III: Network Protocol - Fully Java Driver:**
  + The type III driver is completely written in Java. It is similar to the 3-tier approach to access the database. It helps to send the JDBC method calls to an intermediate server. The intermediate server communicates with the database on behalf of JDBC. The application server converts the JDBC calls either directly or indirectly to the database protocol which is vendor-specific.
  + This approach does not increase the efficiency of architecture and it is costlier, due to this most of the database vendors don’t choose this driver. You need to have good knowledge about the application server for using this approach since the application server is used here.



* **Type IV: Thin Driver - Fully Java Driver**
  + Type IV driver is directly implemented and it directly converts JDBC calls into vendor-specific database protocol. Most of the JDBC Drivers used today are type IV drivers.
  + It is platform-independent since it is written fully in Java. It can be installed inside the Java Virtual Machine(JVM) of the client, so there is no need of installing any software on the client or server side. This drive architecture is having all the logic to communicate directly with the database in a single driver.
  + It provides better performance compared to other driver types. It permits easy deployment. Nowadays, this driver is developed by the database vendor itself so that programmers can use it directly without any dependencies on other sources.



### 12. What are difference between ResultSet and RowSet?

| **ResultSet** | **RowSet** |
| --- | --- |
| ResultSet cannot be serialized as it handles the connection to the database. | RowSet is disconnected from the database so it can be serialized. |
| By default, ResultSet object is non-scrollable and non-updatable. | By default, the RowSet object is scrollable and updatable. |
| ResultSet object is not a JavaBean object. | RowSet object is a JavaBean object. |
| ResultSet is returned by the executeQuery() method of Statement interface | Rowset extends the ResultSet interface and it is returned by calling the RowSetProvider.newFactory().createJdbcRowSet() method. |
| It is difficult to pass ResultSet from one class to another class as it has a connection with the database. | It is easier to pass RowSet from one class to another class as it has no connection with the database. |

### 13. Explain the types of ResultSet.

ResultSet refers to the row and column data contained in a ResultSet object. The object of ResultSet maintains a cursor pointing to the current row of data in the result set.  
There are three types of ResultSet which have constants to control the movement of the cursor in backward, forward, and in a particular row. If we do not declare any ResultSet, then by default TYPE\_FORWARD\_ONLY will be called.

* **ResultSet.TYPE\_FORWARD\_ONLY**: Using this, the cursor can only move forward from start to end in the result set.
* **ResultSet.TYPE\_SCROLL\_INSENSITIVE**: Using this, the cursor can move in both forward and backward directions. Here, the result set is insensitive to the changes done in the database by others, that occur after the result set was created.
* **ResultSet.TYPE\_SCROLL\_SENSITIVE**: Using this, the cursor can move in forward and backward direction, and the result set is sensitive to changes made to the database by others, that occur after the result set was created.

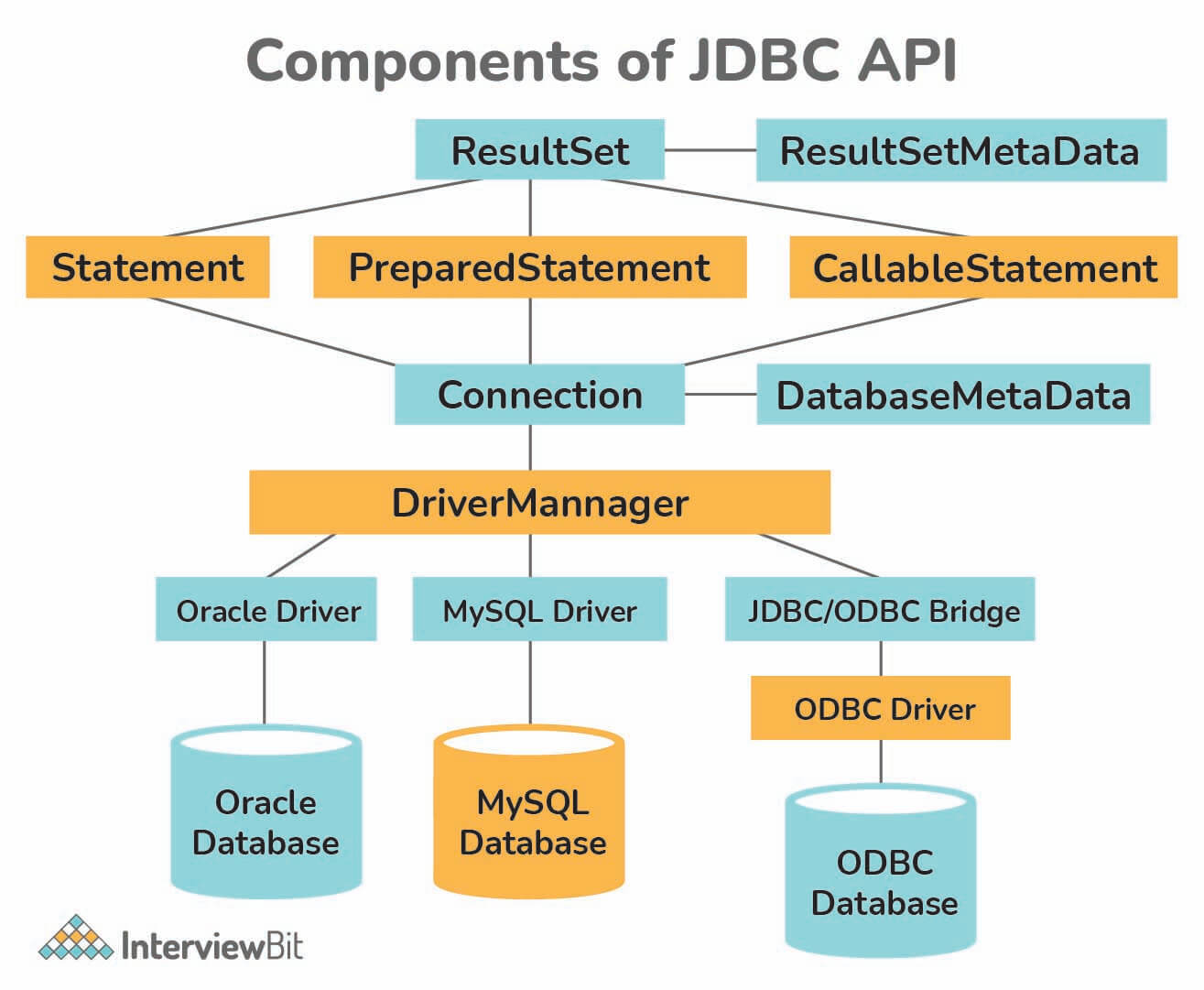
### 14. Explain JDBC API components.

The java.sql package contains different interfaces and classes for JDBC API. They are:  
  
**Interfaces**:

* **Connection**: The object of Connection is created by using getConnection() method of DriverManager class. DriverManager is the factory for connection.
* **Statement**: The object of the Statement is created by using createStatement() method of the Connection class. The Connection interface is the factory for Statement.
* **PreparedStatement**: The PreparedStatement object is created by using prepareStatement() method of Connection class. It is used for executing the parameterized query.
* **ResultSet:** The ResultSet object maintains a cursor pointing to a table row. At first, the cursor points before the first row. The executeQuery() method of the Statement interface returns the object of ResultSet.
* **ResultSetMetaData**: The ResultSetMetaData interface object contains the details about the data(table) such as number of columns, name of the column, column type etc. The getMetaData() method of ResultSet returns the ResultSetMetaData object.
* **DatabaseMetaData**: It is an interface that has methods to get metadata of a database, like name of the database product, version of database product, driver name, name of the total number of views, name of the total number of tables, etc. The getMetaData() method that belongs to Connection interface returns the DatabaseMetaData object.
* **CallableStatement**: CallableStatement interface is useful for calling the stored procedures and functions. We can have business logic on the database through the usage of stored procedures and functions, which will be helpful for the improvement in the performance as these are pre-compiled. The prepareCall() method that belongs to the Connection interface returns the object of CallableStatement.

**Classes:**

* **DriverManager**: It pretends to be an interface between the user and drivers. DriverManager keeps track of the available drivers and handles establishing a connection between a database and the relevant driver. It contains various methods to keep the interaction between the user and drivers.
* **BLOB:** BLOB stands for Binary Large Object. It represents a collection of binary data such as images, audio, and video files, etc., which is stored as a single entity in the DBMS(Database Management System).
* **CLOB:** CLOB stands for Character Large Object. This data type is used by multiple database management systems to store character files. It is the same as BLOB except for the difference, instead of binary data, CLOB represents character stream data such as character files, etc.



### 15. What are the types of JDBC statements?

Statements are useful for sending SQL commands to the database and receiving data from the database. There are three types of statements in JDBC. They are:

* **Statement:** It is the factory for ResultSet. It is used for general-purpose access to the database by executing the static SQL query at runtime. Example:

Statement st = conn.createStatement( );

ResultSet rs = st.executeQuery();

* **PreparedStatement:**It is used when we need to give input data to the query at runtime and also if we want to execute SQL statements repeatedly. It is more efficient than a statement because it involves the pre-compilation of SQL. Example:

String SQL = "Update item SET limit = ? WHERE itemType = ?";

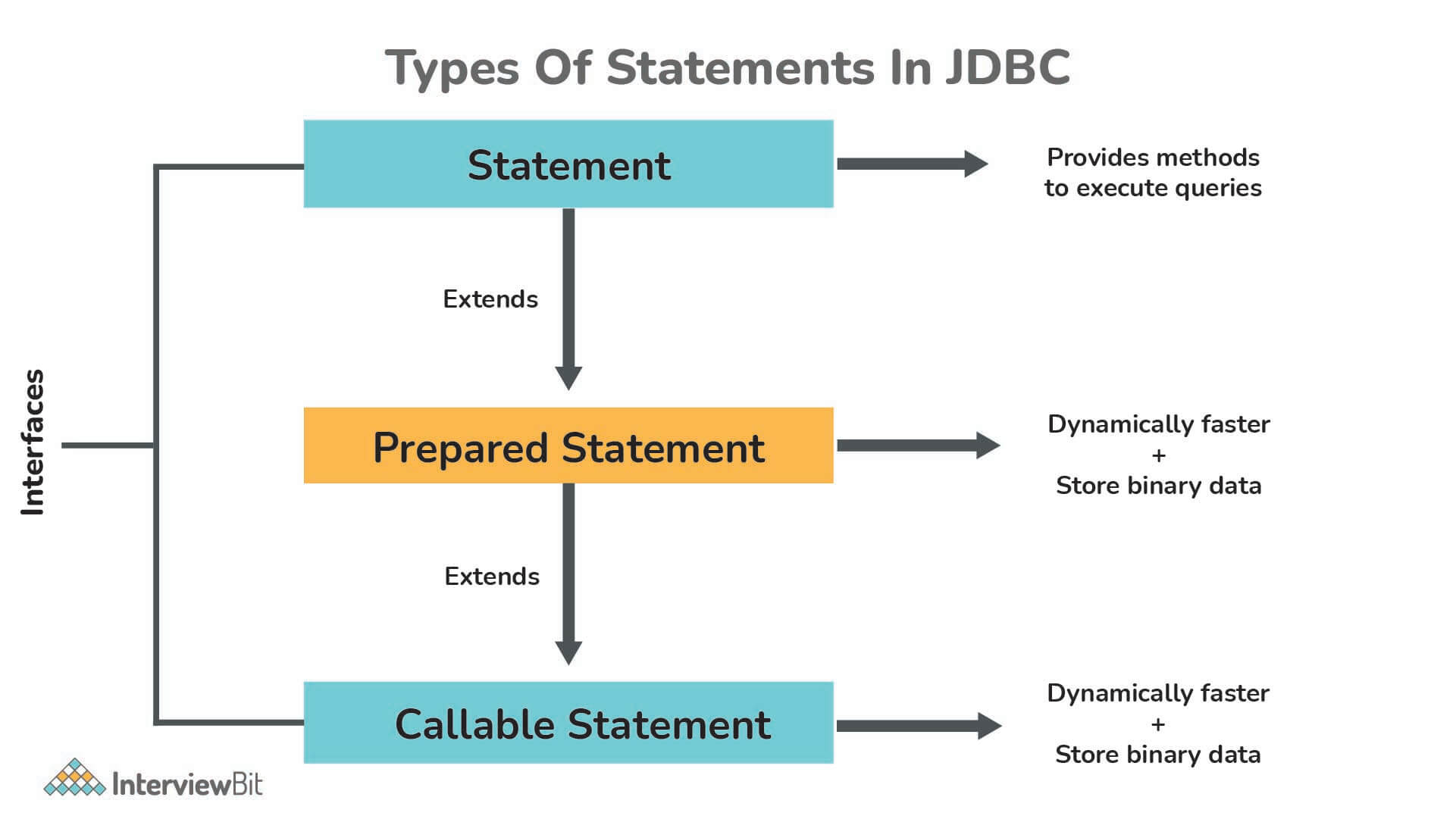
PreparedStatement ps = conn.prepareStatement(SQL);

ResultSet rs = ps.executeQuery();

* **CallableStatement:** It is used to call stored procedures on the database. It is capable of accepting runtime parameters. Example:

CallableStatement cs = con.prepareCall("{call SHOW\_CUSTOMERS}");

ResultSet rs = cs.executeQuery();



### 16. Explain JDBC Batch processing.

* Batch processing is the process of executing multiple SQL statements in one transaction. For example, consider the case of loading data from CSV(Comma-Separated Values) files to relational database tables. Instead of using Statement or PreparedStatement, we can use Batch processing which executes the bulk of queries in a single go for a database.
* *Advantages of Batch processing:*
  + It will reduce the communication time and improves performance.
  + Batch processing makes it easier to process a huge amount of data and consistency of data is also maintained.
  + It is much faster than executing a single statement at a time because of the fewer number of database calls.
* *How to perform Batch processing?*  
  To perform Batch processing, addBatch() and executeBatch() methods are used. These 2 methods are available in the Statement and PreparedStatement classes of JDBC API.

### 17. What is the difference between Statement and PreparedStatement?

| **Statement** | **PreparedStatement** |
| --- | --- |
| The query is compiled every time we run the program. | The query is compiled only once. |
| It is used in the situation where we need to run the SQL query without providing parameters at runtime. | It is used when we want to give input parameters to the query at runtime. |
| Performance is less compared to PreparedStatement. | Provides better performance than Statement, as it executes the pre-compiled SQL statements. |
| It is suitable for executing DDL statements such as CREATE, ALTER, DROP and TRUNCATE. | It is suitable for executing DML statements such as INSERT, UPDATE, and DELETE. |
| It cannot be used for storing/retrieving images and files in the database. | It can be used for storing/retrieving images and files in the database. |
| It executes static SQL statements. | It executes pre-compiled SQL statements. |
| Less secured as it enforces SQL injection. | More secured as they use bind variables, which can prevent SQL injection. |

### 18. What is DataSource in JDBC? What are its benefits?

* DataSource is an interface defined in javax.sql package and is used for obtaining the database connection. It can be used as a good alternative for a DriverManager class as it allows the details about the database to your application program.
* A driver that is accessed through a DataSource object, does not register itself with the DriverManager. Instead, a DataSource object is retrieved through a lookup operation and then it can be used to create a Connection object.
* **Benefits of DataSource:**
  + Caching of PreparedStatement for faster processing
  + ResultSet maximum size threshold
  + Logging features
  + Connection timeout settings
  + Connection Pooling in servlet container using the support of JNDI registry.

### 19. Explain the difference between execute(), executeQuery() and executeUpdate() methods in JDBC.

| **execute()** | **executeQuery()** | **executeUpdate()** |
| --- | --- | --- |
| It can be used for any SQL statements. | It is used to execute SQL Select queries. | It is used to execute the SQL statements such as Insert/Update/Delete which will update or modify the database data. |
| It returns the boolean value TRUE if the result is a ResultSet object and FALSE when there is no ResultSet object. | It returns the ResultSet object which contains the data retrieved by the SELECT statement. | It returns an integer value which represents the number of affected rows where 0 indicates that the query returns null. |
| Used for executing both SELECT and non-SELECT queries. | Used for executing only the SELECT Query. | Used for executing only a non-SELECT query. |